Appendix: Introduction to MATLAB

This appendix introduces the reader to programming with the software package MATLAB. It is assumed that the reader has had previous experience with a high-level programming language and is familiar with the techniques of writing loops, branching using logical relations, calling subroutines, and editing. These techniques are directly applicable in the windows-type environment of MATLAB.

MATLAB is a mathematical software package based on matrices. The package consists of an extensive library of numerical routines, easily accessed two- and three-dimensional graphics, and a high-level programming format. The ability to implement and modify programs quickly makes MATLAB an appropriate format for exploring and executing the algorithms in this textbook.

The reader should work through the following tutorial introduction to MATLAB (MATLAB commands are in `typewriter` type). The examples illustrate typical input and output from the MATLAB Command Window. To find additional information about commands, options, and examples, the reader is urged to make use of the on-line help facility and the Reference and User's Guides that accompany the software.

Arithmetic Operations

`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	Division
`^`	Power
`pi, e, i`	Constants

```
Ex. >>(2+3*pi)/2
    ans =
         5.7124
```

Built-in Functions

Below is a short list of some of the functions available in MATLAB. The following example illustrates how functions and arithmetic operations are combined. Descriptions of other available functions may be found by using the on-line help facility.

```
    abs(#)    cos(#)    exp(#)    log(#)    log10(#)    cosh(#)
    sin(#)    tan(#)    sqrt(#)   floor(#)  acos(#)     tanh(#)
Ex. >>3*cos(sqrt(4.7))
    ans =
         -1.6869
```

The default format shows approximately five significant decimal figures. Entering the command `format long` will display approximately 15 significant decimal figures.

```
Ex. >>format long
    3*cos(sqrt(4.7))
    ans =
         -1.68686892236893
```

Assignment Statements

Variable names are assigned to expressions by using an equal sign.

```
Ex. >>a=3-floor(exp(2.9))
    a=
         -15
```

A semicolon placed at the end of an expression suppresses the computer echo (output).

```
Ex. >>b=sin(a);              Note. b was not displayed.
    >>2*b^2
    ans=
         0.8457
```

Defining Functions

In MATLAB the user can define a function by constructing an M-file (a file ending in .m) in the M-file Editor/Debugger. Once defined, a user-defined function is called in the same manner as built-in functions.

Ex. Place the function $\text{fun}(x) = 1 + x - x^2/4$ in the M-file `fun.m`. In the Editor/Debugger one would enter the following:

```
function y=fun(x)
y=1+x-x.^2/4;
```

We will explain the use of ".^" shortly. Different letters could be used for the variables and a different name could be used for the function, but the same format would have to be followed. Once this function has been saved as an M-file named `fun.m`, it can be called in the MATLAB Command Window in the same manner as any function.

```
>>cos(fun(3))
ans=
     -0.1782
```

A useful and efficient way to evaluate functions is to use the `feval` command. This command requires that the function be called as a string.

```
Ex. >>feval('fun',4)
    ans=
         1
```

Matrices

All variables in MATLAB are treated as matrices or arrays. Matrices can be entered directly:

```
Ex. >>A=[1 2 3;4 5 6;7 8 9]
    A=
         1 2 3
         4 5 6
         7 8 9
```

Semicolons are used to separate the rows of a matrix. Note that the entries of the matrix must be separated by a single space. Alternatively, a matrix can be entered row by row.

```
Ex. >>A=[1 2 3
         4 5 6
         7 8 9]
    A =
         1 2 3
         4 5 6
         7 8 9
```

Matrices can be generated using built-in functions.

```
Ex. >>Z=zeros(3,5);        creates a 3 × 5 matrix of zeros
    >>X=ones(3,5);         creates a 3 × 5 matrix of ones
    >>Y=0:0.5:2            creates the displayed 1 × 5 matrix
    Y=
    0 0.5000 1.0000 1.5000 2.0000
```

```
>>cos(Y)                creates a 1 × 5 matrix by taking the
                        cosine of each entry of Y
ans=
     1.0000 0.8776 0.5403 0.0707 -0.4161
```

The components of matrices can be manipulated in several ways.

```
Ex. >>A(2,3)                select a single entry of A
    ans=
         6
    >>A(1:2,2:3)            select a submatrix of A
    ans=
         2 3
         5 6
    >>A([1 3],[1 3])        another way to select a submatrix of A
    ans=
         1 3
         7 9
    >>A(2,2)=tan(7.8);      assign a new value to an entry of A
```

Additional commands for matrices can be found by using the on-line help facility or consulting the documentation accompanying the software.

Matrix Operations

`+`	Addition
`-`	Subtraction
`*`	Multiplication
`^`	Power
`'`	Conjugate transpose

```
Ex. >>B=[1 2;3 4];
    >>C=B'                  C is the transpose of B
    C=
         1 3
         2 4
    >>3*(B*C)^3             3(BC)^3
    ans=
         13080 29568
         29568 66840
```

Array Operations

One of the most useful characteristics of the MATLAB package is the number of functions that can operate on the individual elements of a matrix. This was demonstrated

earlier when the cosine of the entries of a 1×5 matrix was taken. The matrix operations of addition, subtraction, and scalar multiplication already operate elementwise, but the matrix operations of multiplication, division, and power do not. These three operations can be made to operate elementwise by preceding them with a period: `.*`, `./`, and `.^`. It is important to understand how and when to use these operations. Array operations are crucial to the efficient construction and execution of MATLAB programs and graphics.

```
Ex. >>A=[1 2;3 4];
    >>A^2                 produces the matrix product AA
    ans=
         7 10
         15 22
    >>A.^2                squares each entry of A
    ans=
         1 4
         9 16
    >>cos(A./2)           divides each entry of A by 2, then takes
                          the cosine of each entry
    ans=
         0.8776  0.5403
         0.0707 -0.4161
```

Graphics

MATLAB can produce two- and three-dimensional plots of curves and surfaces. Options and additional features of graphics in MATLAB can be found in the on-line facility and the documentation accompanying the software.

The `plot` command is used to generate graphs of two-dimensional functions. The following example will create the plot of the graphs of $y = \cos(x)$ and $y = \cos^2(x)$ over the interval $[0, \pi]$.

```
Ex. >>x=0:0.1:pi;
    >>y=cos(x);
    >>z=cos(x).^2;
    >>plot(x,y,x,z,'o')
```

The first line specifies the domain with a step size of 0.1. The next two lines define the two functions. Note that the first three lines all end in a semicolon. The semicolon is necessary to suppress the echoing of the matrices `x`, `y`, and `z` on the command screen. The fourth line contains the plot command that produces the graph. The first two terms in the plot command, `x` and `y`, plot the function $y = \cos(x)$. The third and fourth terms, `x` and `z`, produce the plot of $y = \cos^2(x)$. The last term, `'o'`, results in o's being plotted at each point (x_k, z_k) where $z_k = \cos^2(x_k)$.

In the third line the use of the array operation ".^" is critical. First the cosine of each entry in the matrix x is taken, and then each entry in the matrix cos(x) is squared using the .^ command.

The graphics command fplot is a useful alternative to the plot command. The form of the command is fplot('name',[a,b],n). This command creates a plot of the function name.m by sampling n points in the interval $[a, b]$. The default number for n is 25.

Ex. >>fplot('tanh',[-2,2]) plots $y = \tanh(x)$ over $[-2, 2]$

The plot and plot3 commands are used to graph parametric curves in two- and three-dimensional space, respectively. These commands are particularly useful in the visualization of the solutions of differential equations in two and three dimensions.

Ex. The plot of the ellipse $c(t) = (2\cos(t), 3\sin(t))$, where $0 \leq t \leq 2\pi$, is produced with the following commands:

```
>>t=0:0.2:2*pi;
>>plot(2*cos(t),3*sin(t))
```

Ex. The plot of the curve $c(t) = (2\cos(t), t^2, 1/t)$, where $0.1 \leq t \leq 4\pi$, is produced with the following commands:

```
>>t=0.1:0.1:4*pi;
>>plot3(2*cos(t),t.^2,1./t)
```

Three-dimensional surface plots are obtained by specifying a rectangular subset of the domain of a function with the meshgrid command and then using the mesh or surf commands to obtain a graph. These graphs are helpful in visualizing the solutions of partial differential equations.

Ex.
```
>>x=-pi:0.1:pi;
>>y=x;
>>[x,y]=meshgrid(x,y);
>>z=sin(cos(x+y));
>>mesh(z)
```

Loops and Conditionals

Relational Operators

==	Equal to
~=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

Logical Operators

`~`	Not	(complement)
`&`	And	(true if both operands are true)
`|`	Or	(true if either or both operands are true)

Boolean Values

`1`	True
`0`	False

The `for`, `if`, and `while` statements in MATLAB operate in a manner analogous to their counterparts in other programming languages. These statements have the following basic form:

```
for (loop-variable = loop-expression)
        executable-statements
end

if (logical-expression)
        executable-statements
else (logical- expression)
        executable-statements
end

while (while-expression)
        executable-statements
end
```

The following example shows how to use nested loops to generate a matrix. The following file was saved as a M-file named `nest.m`. Typing `nest` in the MATLAB Command Window produces the matrix `A`. Note that, when viewed from the upper-left corner, the entries of the matrix `A` are the entries in Pascal's triangle.

```
Ex. for i=1:5
      A(i,1)=1;A(1,i)=1;
    end
    for i=2:5
      for j=2:5
        A(i,j)=A(i,j-1)+A(i-1,j);
      end
    end
    A
```

The `break` command is used to exit from a loop.

```
Ex. for k=1:100
      x=sqrt(k);
      if ((k>10)&(x-floor(x)==0))
        break
      end
    end
```

```
    k
```

The `disp` command can be used to display text or a matrix.

```
Ex. n=10;
    k=0;
    while k<=n
      x=k/3;
      disp([x x^2 x^3])
      k=k+1;
    end
```

Programs

An efficient way to construct programs is to use user-defined functions. These functions are saved as M-files. These programs allow the user to specify the input and output parameters. They are easily called as subroutines in other programs. The following example allows one to visualize the effects of moding out Pascal's triangle with a prime number. Type the following function in the MATLAB Editor/Debugger and then save it as an M-file named `pasc.m`.

```
Ex. function P=pasc(n,m)
    %Input  - n is the number of rows
    %       - m is the prime number
    %Output - P is Pascal's triangle

    for j=1:n
      P(j,1)=1;P(1,j)=1;
    end
    for k=2:n
      for j=2:n
        P(k,j)=rem(P(k,j-1)+P(k-1,j),m);
      end
    end
```

Now in the MATLAB Command Window enter `P=pasc(5,3)` to see the first five rows of Pascal's triangle mod 3. Or try `P=pasc(175,3);` (note the semicolon) and then type `spy(P)` (generates a sparse matrix for large values of n).

Conclusion

At this point the reader should be able to create and modify programs based on the algorithms in this textbook. Additional information on commands and information regarding the use of MATLAB on your particular platform can be found in the on-line help facility or in the documentation accompanying the software.

Numerical Methods Using Matlab, 4th Edition, 2004

John H. Mathews and Kurtis K. Fink

ISBN: 0-13-065248-2

Prentice-Hall Inc.
Upper Saddle River, New Jersey, USA
http://vig.prenhall.com/